4 pages/down

SPCA ZUSAMMENFASSUNG

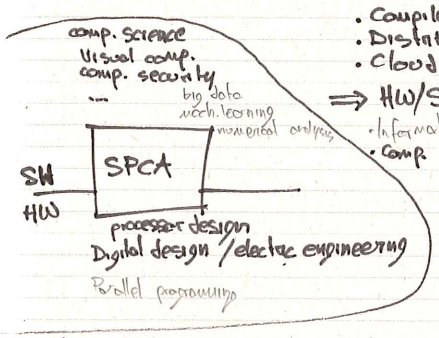
Simone Guggiari
Zusammenfassung
HS 2015 - ETHZ

1. INTRO

- what makes progr. go fast/slow
  → write fast/correct/good syst. code
  → C + Assembly

related: CS 15-213 Carnegie Mellon University COMPUTER SYSTEMS - Bryant

1.1 - What is systems?



- OS
- Databases
- Networking / routing
- Compiler (design & implement)
- Distributed syst
- Cloud computing

⇒ HW/SW boundary
- Information Entitled systems
- Comp Architecture

Motivation: too much focus on abstraction → asymptotic (complexity) data types (obj/contracts)
→ limitations: don't survive contact w/ reality (due to bugs & underlying implement)

⇒ GOALS: more effective programmers
- find/eliminate bugs efficiently
- tune program performance

1.2 FIVE REALITIES

#1. Int ≠ integers
float ≠ reals

ex: $x^2 \geq 0$? float yes! int overflow ($50k^2$)

$(x+y)+z \overset{?}{=} x+(y+z)$
int yes
float $1e20 + (-1e20 + \pi) = ?$

* code security example

Comp. Arithmetic → different properties than mathematical arithm.

⚠ finiteness of representation

INT: ring properties $a+b=b+a$ $a+(b+c)=(a+b)+c$ $a(b+c)=ab+ac$ (commutativity / associativity / distributivity)

FLOAT: ordering properties (no ring!) (monotonicity, values of signs)

#2. you must know assembly
→ probably will never write
→ understanding is key for optimization / machine lvl execution
- behavior in presence of bugs (high lvl lang breaks down)
- performance (optimization by compiler) + sources of inefficiency
- system SOFTWARE compiler produces machine code / OS manages process states
- creating / fighting malware

* time stamp counter (clk counter) ⇒ GCC's asm (use it to integrate assembly in .c file)

#3. Memory matters. RAM is unrealistic
mem not unbounded: allocated & managed
⚠ mem. referencing bugs (they happen after hrs)
mem. performance not uniform (→ adapt prog to mem. hierarchy to speed up)
cache + VM

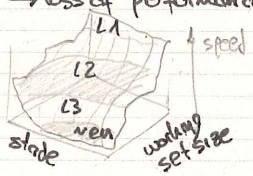
volatile: tell compiler not to optimize anything that has to do with variable you interface with HW (Oop reordering, w-loop busy wait)

C/C++: no mem protection
- out of bound array
- invalid ptr
- abuse malloc/free

} nasty bug (might not show, or just after much hrs) coupled obj might be uncleaned

ex matrix mult. bcp → loss of performance (hierarchical mem organiz. cache 160x!

memory mountain



#4. performance ≠ asympt. complexity
→ const. matter
op. count ≠ performance (optimize on multiple lvl: • alg • data repr • procedures • loops
⇒ identify bottle necks ⚠ improve performance (optimize) without destroying modularity & flexibility
• blocking/tiling/loop unrolling/array scalarization/instr. scheduling
→ less reg spills, less L1/L2 cache miss / less TLB misses

#5. comp. don't just execute progr.
→ data in/out (input/output devices)
→ network communic • concurrent comp ...
* modern SoC (system on a chip)

1.3 what you should know

strong-typing = specify each type (C is weakly-typed)

→ bin. representation
bistable elements (reliable on noisy wires)
encoding: first digit must not be 0 in c! (decimal)

bin: 00000000 - 11111111
Hex: 00 - ff
dec: 0 - 255

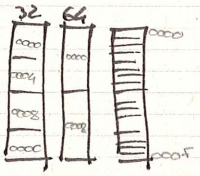
byte-oriented mem-org:
→ refer to VM address (conceptually, very large array of bytes)
→ but in reality a hyerarchy of mem types

OS provide address space to process (disjunct)
compiler + runtime syst control allocation → where obj should be stored

Words: size of int → often 32bit (4bytes) ⇒ 4 GB adresses
often too small → use 64 bit ⇒ 1.8×10^{19} bytes
or 48 ⇒ 256 TB

multiple data format: fraction 4/multiple of word size

Word-oriented mem:
address → byte location



differ 4/8 bytes from word

Byte-ordering:

BIG-endian (LSB has bigger)
LITTLE-endian (start from LSB) (LSB has smaller)

ex 0x01234567

Reading byte-reversal listings
text representation of machine code (endian)
deciphering: ex 0x12ab = ab 12 00 00

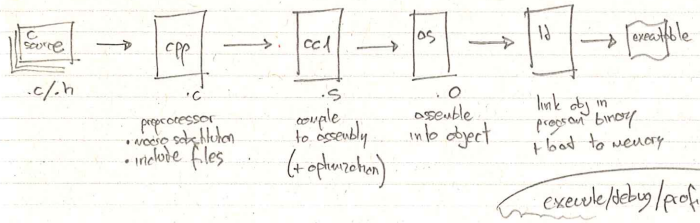
print byte-repr of data (cast ptr to unsigned char)
%p print ptr
%x hex

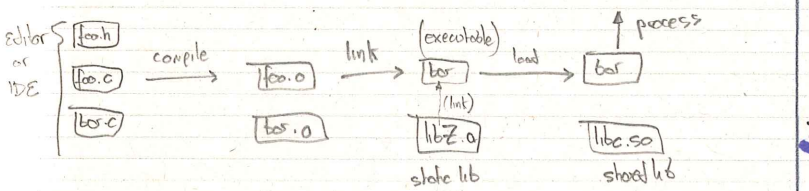
LINUX: 32bit Intel x86 is LITTLE-end

Boolean: closed switch = 1 (does ⊟ pass?)
open = 0

SCHEMES

GNU TOOLCHAIN





Int ranges

	U (unsigned)	T (signed, 2's complement)
conversion	$\sum_{i=0}^{w-1} x_i \cdot 2^i$	$-(x_{w-1})2^{w-1} + \sum_{i=0}^{w-2} x_i \cdot 2^i$
min	$0 = 0x0000 = 0000\ldots$	$-2^{w-1} = 0x8000 = 1000\ldots$ $= -32768\ (w=16)$
max	$2^w - 1 = 0xFFFF = 1111$ (65535)	$2^{w-1} - 1 = 0x7FFF = 0111111$ (32767)
-1	N/A	$0xFFFF = 111\ldots$
0	$0 = 0x0000 = 0000\ldots$	$0 = 0x0000 = 0000\ldots$

observations $x_{w-1}\ y_{w-2} \cdots x_1 x_0$

$|U_{max}| = 2|T_{min}| - 1$

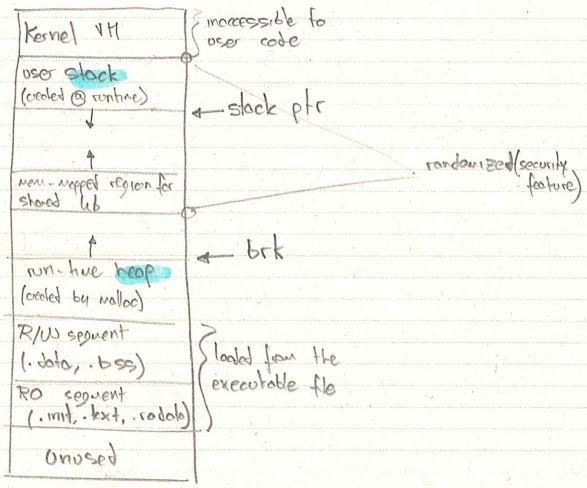
$= 2|T_{max}| + 1$

$|T_{min}| = |T_{max}| + 1$

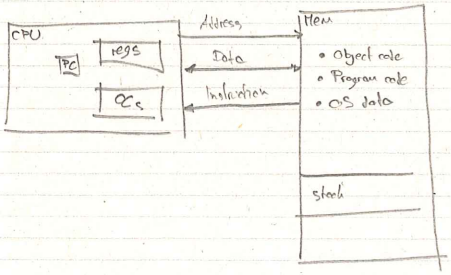
2's compl: $\sim x + 1 = -x$

$\sim x + x = -1 = 1111$

stack

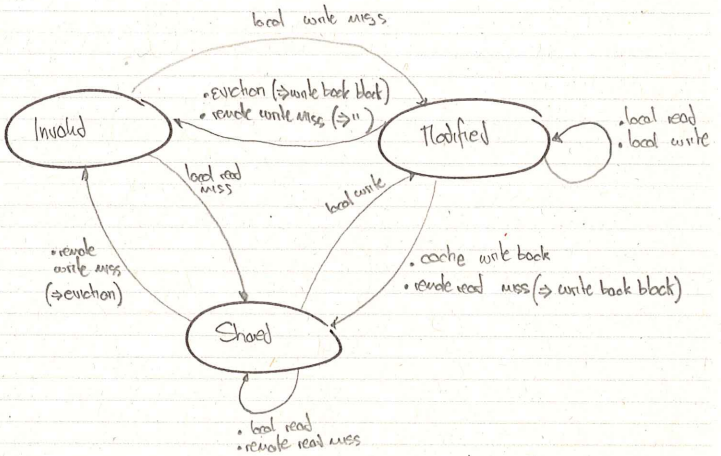


architecture machine code

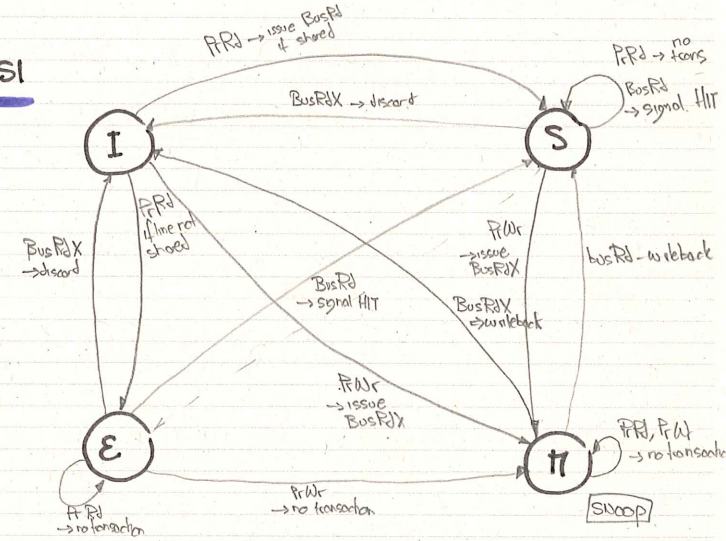


dec	hex	bin	pow
0	0	0000	1
1	1	0001	2
2	2	0010	4
3	3	0011	8
4	4	0100	16
5	5	0101	32
6	6	0110	64
7	7	0111	128
8	8	1000	256
9	9	1001	512
10	A	1010	1024
11	B	1011	2048
12	C	1100	4096
13	D	1101	8192
14	E	1110	16384
15	F	1111	32768

MSI



MESI

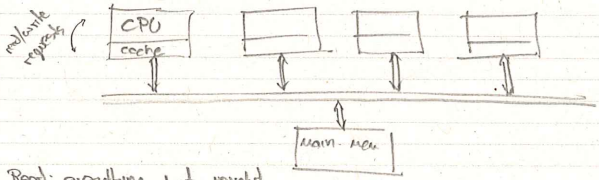


Modified: cache is different from mem → duty bit, needed to write back to mem

Exclusive: present only in current cache, but it's clean (matches main mem)

Shared: might be present in other caches, and it's clean (can evict without writeback)

Invalid: unused (not present in cache) → to be fetched from mem



Read: everything but invalid

Write: Modified/exclusive.

If shared, all others must be invalidated (RFO - request for ownership)